

DawnSci Eclipse Project

- An open source not for profit project*
- On GitHub **'DawnScience'***
- Diamond Light Source Ltd. and the ESRF are largely publically funded research facilities*





Science Working Group

science.eclipse.org



- Charter including a vision for the future
- Contributions from DAWN and NiCE being made – we will look at DAWNSci
- Presentations in the US (2012) and last year, Germany (2013)
- Eclipse Foundation investing in the group
 - Git / Jenkins / Marketing

Who

- Diamond Light Source
- ORNL
- IBM
- IFP Energies Nouvelles
- TECH'Advantage
- MARINTEK / Iteima
- Clemson University
- Lablicate
- Uppsala University
- Kichwa Coders
- The Facility for Rare Isotope Beams at Michigan State University

...What we said/promised last time...

DAWNSCI Eclipse Project Phased Delivery

- **Phase 1 2014/2015** - *Definition of long term APIs and Reference Implementation*
 - HDF5 Loading / Saving API
 - Description of data and some mathematical operations
 - Plotting interface (only) based on data description
 - Slicing interface description
 - Examples of how to use API and reference implementation (binary)
- **Phase 2 2016** – *Release of concrete implementation(s)*
 - TBD

**So this presentation will show you
how to make use of that so far released...**

Diamond Light Source - *RCP Based Projects*

Actively (each ~10s developers) developed at DLS

- Generic Data Acquisition
 - Client in RCP server CORBA
- Data Analysis Workbench
 - That you already know and love
- Control Systems Studio
 - Community active on many sites
 - DESY, NSLS2, KEK, ITER



Diamond Light Source - *Data Analysis Group*

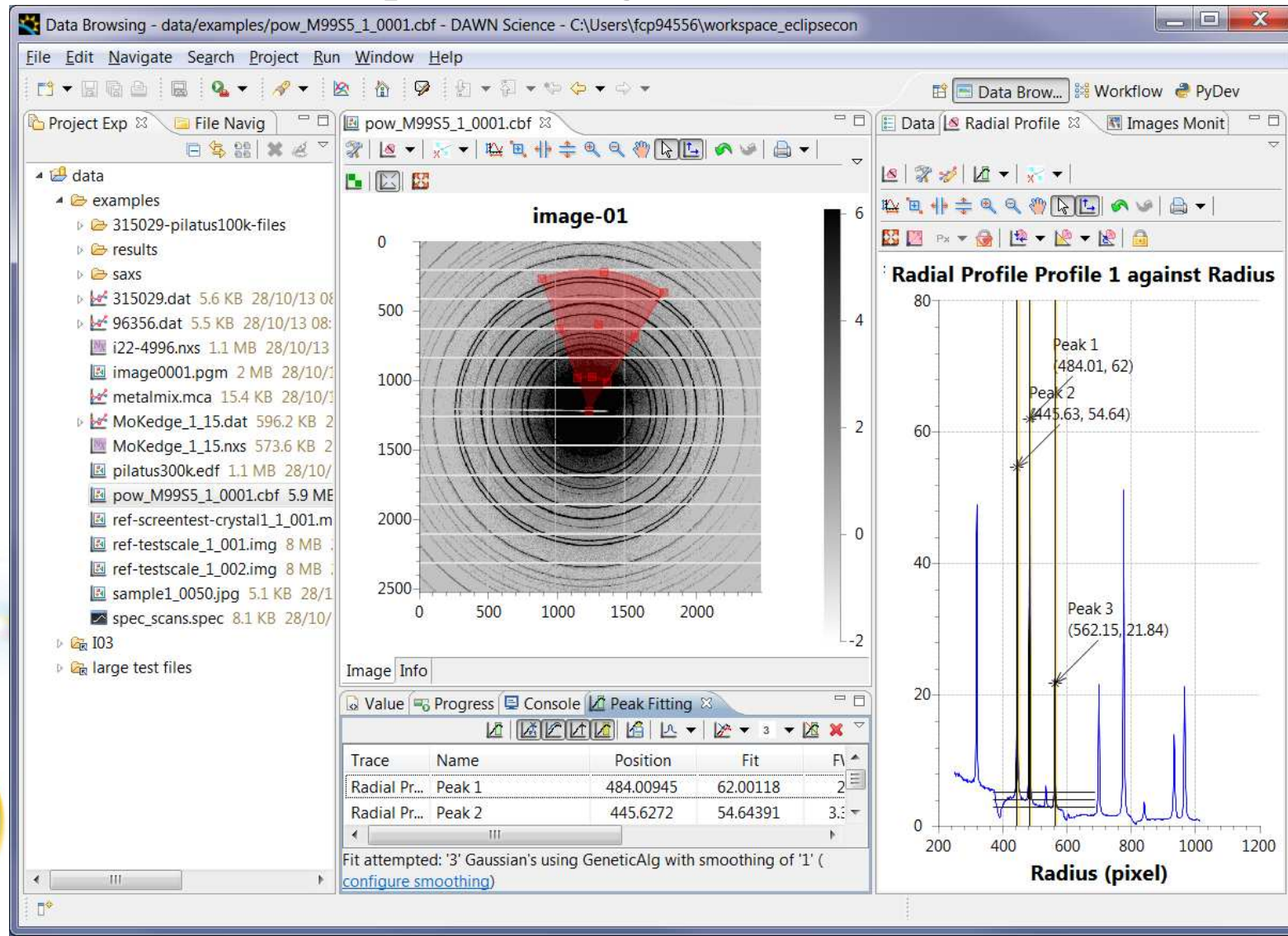
Team	Mostly Doing
DAWN	DAWN - RCP front end for Data Analysis. Support of third party code for Data Analysis. Help/support with python scripts. DAWN includes Numpy.
DIALS	Working in the crystallography area producing new software and running automatic pipelines (cluster) for processing data produced during data collections.
Tomography	Working with scientists to support visualization software. Responsible for reconstruction pipelines running on GPU clusters
Web	ISPyB, ICAT, SyncLink and SyncWeb productions allowing users to interact with experimental data remotely.
Spectroscopy / 1D tools	Developing tools for 1D analysis like peak fitting. Python interactivity a key requirement.

DAWN Lots of Visual Tools

- For images
 - Line, Box, Sector integration
 - Diffraction image interpretation, line profile for 'D-spacing'
 - Color mapping / Histogramming
 - Pixel Information and region control
- For XY Graphs
 - Peak Fitting and Line Fitting
 - Derivative and other functions, including user defined
 - Scientific tools
 - XAFS Analysis Tool
 - SAXS
- Use of eclipse architecture, extension points and pages inside PageBookView.

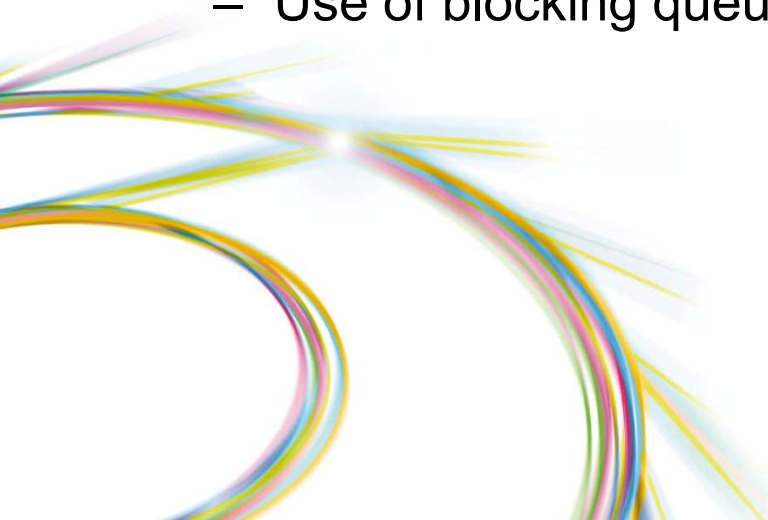
Demonstration — Visual Tools

Example showing various visual tools



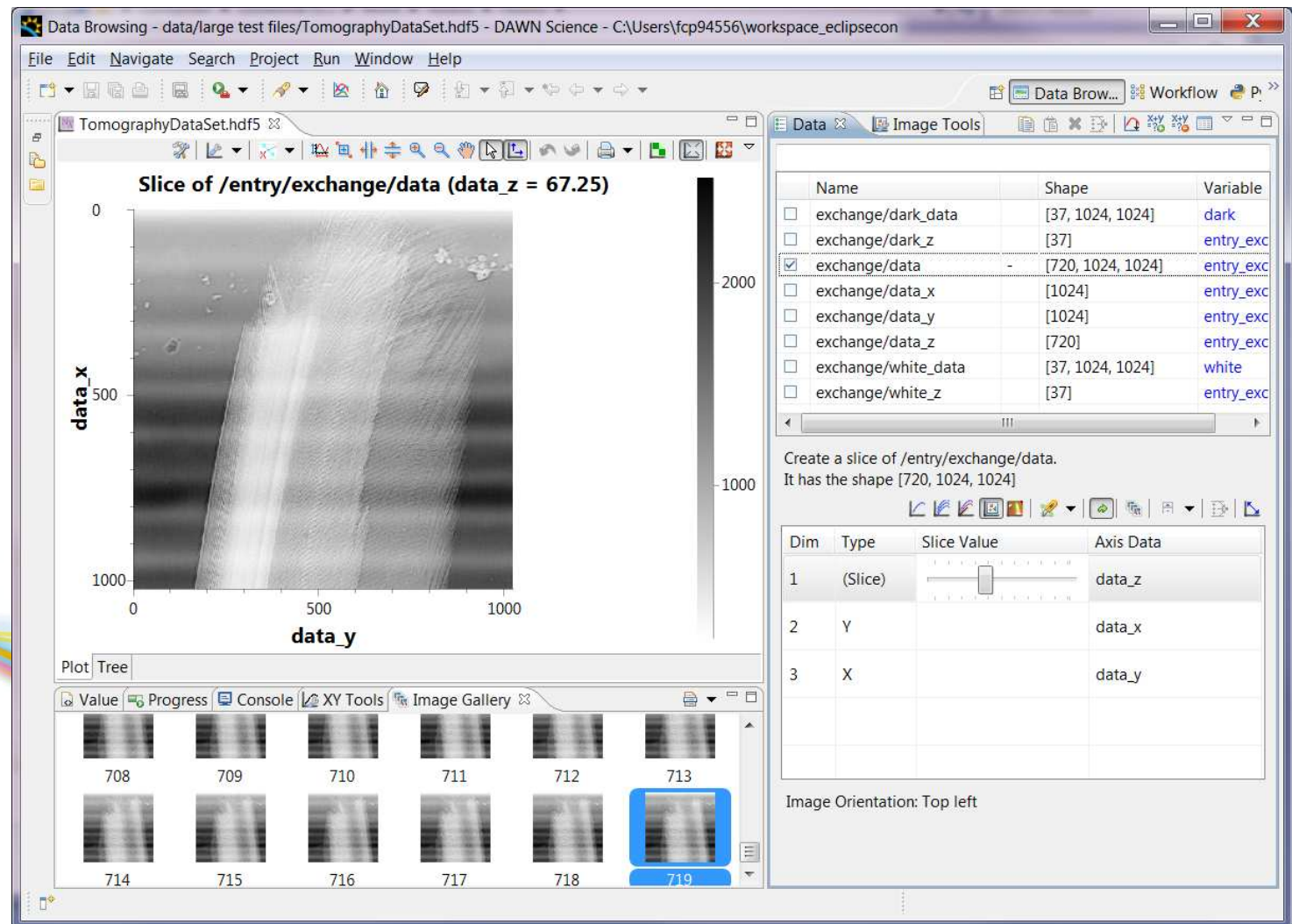
Slicing data

- Cutting through N-dimensional data
 - With an XY plot
 - As an image
 - As a 3D iso-surface
 - Hyper 3D
- Important to run everything concurrently
 - Use of Jobs
 - Use of ordinary threads
 - Use of blocking queues



Demonstration — Slicing and dicing

Example opening a tomography file and slicing it



Introducing The DAWNSci Eclipse Project...

- **Data**
 - **Loading (inc. HDF5)**
 - **Description**
 - **Slicing**
 - **Transformation**
 - **Mathematic**
 - **Metadata**
- **User Interface**
 - **Plotting 1/2/3 D**
 - **Slicing nD**



**Phase 1 is API with a
reference implementation via a p2 site**

<http://www.dawnsi.org/eclipse/getting-started-with-dawnsi>

Example : Loading and Manipulating Data

```
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 3.0000 7.0000
3.0000 1.0000 2.0000 0.0000 2.0000 2.0000 1.0000 3.0000 2.0000 3.0000
2.0000 2.0000 3.0000 2.0000 1.0000 0.0000 3.0000 1.0000 1.0000 3.0000
2.0000 2.0000 1.0000 3.0000 2.0000 1.0000 2.0000 3.0000 4.0000 0.0000
0.0000 0.0000 1.0000 2.0000 0.0000 2.0000 2.0000 0.0000 0.0000 4.0000
1.0000 1.0000 3.0000 3.0000 0.0000 1.0000 0.0000 1.0000 0.0000 2.0000
0.0000 1.0000 0.0000 0.0000 3.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 1.0000 0.0000 0.0000 1.0000 1.0000 1.0000 1.0000
1.0000 2.0000 0.0000 0.0000 2.0000 0.0000 1.0000 1.0000 0.0000 1.0000
2.0000 0.0000 1.0000 1.0000 2.0000 1.0000 0.0000 1.0000 3.0000 2.0000
1.0000 2.0000 1.0000 2.0000 1.0000 1.0000 1.0000 0.0000 3.0000 2.0000
3.0000 1.0000 0.0000 2.0000 1.0000 0.0000 1.0000 0.0000 0.0000 1.0000
1.0000 0.0000 1.0000 0.0000 1.0000 4.0000 0.0000 1.0000 0.0000 0.0000
0.0000 1.0000 1.0000 1.0000 3.0000 1.0000 0.0000 0.0000 0.0000 0.0000
1.0000 1.0000 1.0000 0.0000 1.0000 1.0000 1.0000 0.0000 1.0000 0.0000
2.0000 3.0000 1.0000 0.0000 0.0000 1.0000 0.0000 2.0000 2.0000 1.0000
1.0000 0.0000 0.0000 1.0000 2.0000 0.0000 4.0000 1.0000 1.0000 2.0000
1.0000 1.0000 0.0000 0.0000 1.0000 2.0000 0.0000 3.0000 2.0000 0.0000
0.0000 1.0000 0.0000 2.0000 0.0000 1.0000 0.0000 2.0000 0.0000 2.0000
1.0000 0.0000 0.0000 0.0000 0.0000 1.0000 1.0000 1.0000 0.0000 3.0000
0.0000 0.0000 0.0000 0.0000 0.0000 3.0000 0.0000 1.0000 1.0000 0.0000
2.0000 1.0000 2.0000 0.0000 0.0000 1.0000 0.0000 4.0000 1.0000 0.0000
0.0000 0.0000 0.0000 1.0000 2.0000 3.0000 0.0000 2.0000 3.0000 1.0000
0.0000 3.0000 2.0000 1.0000 5.0000 4.0000 4.0000 5.0000 4.0000 4.0000
1.0000 1.0000 1.0000 1.0000 2.0000 2.0000 0.0000 1.0000 2.0000 3.0000
2.0000 0.0000 1.0000 2.0000 1.0000 3.0000 0.0000 1.0000 1.0000 4.0000
0.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000 1.0000 1.0000 2.0000
1.0000 3.0000 2.0000 1.0000 0.0000 0.0000 1.0000 0.0000 0.0000 2.0000
0.0000 1.0000 1.0000 1.0000 1.0000 0.0000 0.0000 0.0000 1.0000 1.0000
0.0000 0.0000 1.0000 0.0000 2.0000 3.0000 1.0000 0.0000 2.0000 0.0000
3.0000 1.0000 1.0000 0.0000 2.0000 1.0000 1.0000 1.0000 2.0000 5.0000
0.0000 0.0000 4.0000 2.0000 3.0000 2.0000 1.0000 2.0000 3.0000 2.0000
1.0000 2.0000 3.0000 1.0000 1.0000 2.0000 1.0000 2.0000 1.0000 0.0000
1.0000 1.0000 2.0000 1.0000 1.0000 1.0000 0.0000 2.0000 1.0000 2.0000
2.0000 0.0000 2.0000 1.0000 2.0000 2.0000 0.0000 3.0000 1.0000 3.0000
1 0000 1 0000 0 0000 0 0000 2 0000 2 0000 1 0000 0 0000 1 0000 0 0000
```

This example

- Columns of ASCII data
- Separated by tabs
- Read as 1D arrays

Other format with a loader:

srs, dat, flt, gff, mca, csv, xy, xye,
txt, tif, tiff, cbf, img, ciff, mccd, edf,
pgm, cor, bruker, jpg, jpeg, png,
f2d, msk, mib, mar3450, pck3450,
raw, mrc, gz, bz2, zip, h5, hd5,
hdf5, nxs, nexus, hdf, mat

Extension point to add more...

DAWNSCI - ILoaderService

Purpose – load any numerical file format using a low dependency service. Keeps code modular and it is easy to use.

How – Loaders can be contributed by extension point for custom file formats. Many loaders come pre-registered with the service.

Example - Reading and slicing data

```
ILoaderService service = (ILoaderService)Activator.getService(ILoaderService.class);  
IDataHolder holder = service.getData(<file path>, ...);  
IDataset full = holder.getDataset("Column_1");
```

```
// full is the full array of data loaded into memory represented by  
// an object called IDataset. Mathematics and plotting can be done  
// using this object as if it were an array.
```

Show LoaderExamples...

DAWNSCI - ILazyDataset and IDataset

Purpose – represent data without physically loading it into memory, delaying this action until it is required.

How – The LazyDatasetImpl might use HDF5 or directories of files to represent stacks of data.

Example - Slicing data

```
ILazyDataset lz = holder.getLazyDataset("Big"); // Not all loaded, too big
System.out.println("Your data shape is "+ lz.getShape());
```

```
IDataset slice = lz.getSlice(...); // Now it's loaded
IDataset all = holder.getDataset("myData"); // Load everything!!
```

```
// Real slices of the data can be taken from the ILazyDataset and
// expressions written which result value is only evaluated when the
// slice is visualized.
```

Show NumpyExamples and LazyExamples

DAWNSCI - IMetadata

Purpose – to provide access to experimental conditions without loading all the numerical / binary data.

How – A loader may define how to load up header information or other attributes from the file and represent access to these as IMetadata

Example - Metadata

```
IMetadata meta = holder.getMetadata();  
System.out.println("The temperature was "+ meta.getMetaValue("T") ;
```

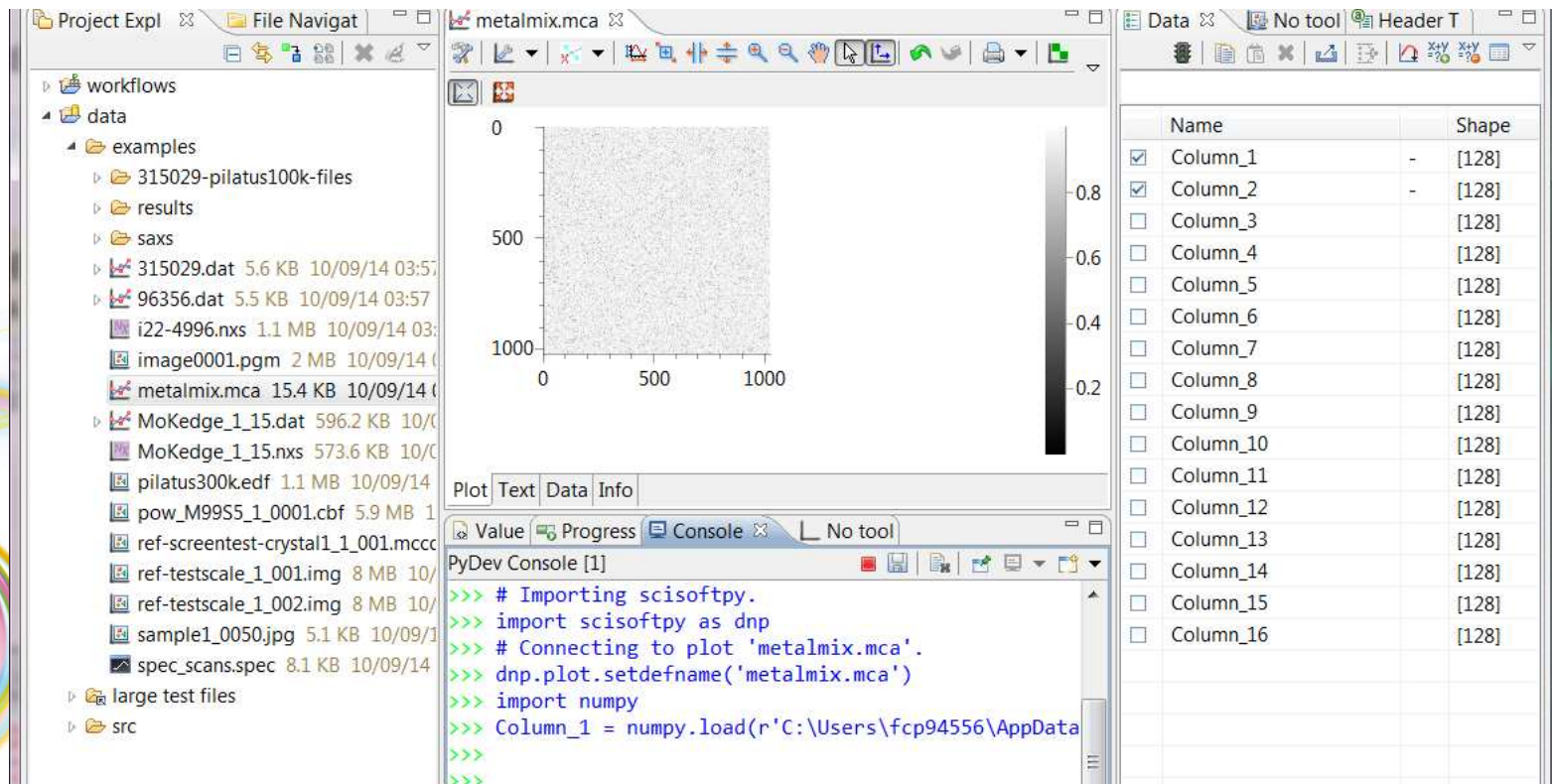
```
IDataset data = holder.getDataset("myData") ;  
meta = holder.getMetadata() ;  
System.out.println("The temperature was "+ meta.getMetaValue("T") ;
```

```
// The meta data can exist for the whole file or for individual  
datasets.
```


DAWNSCI - Python/Numpy

Purpose – to make available data in the GUI to python scripting

How – A flattening service transfers data to and from python. IDataset appears as a numpy array in a seamless way.



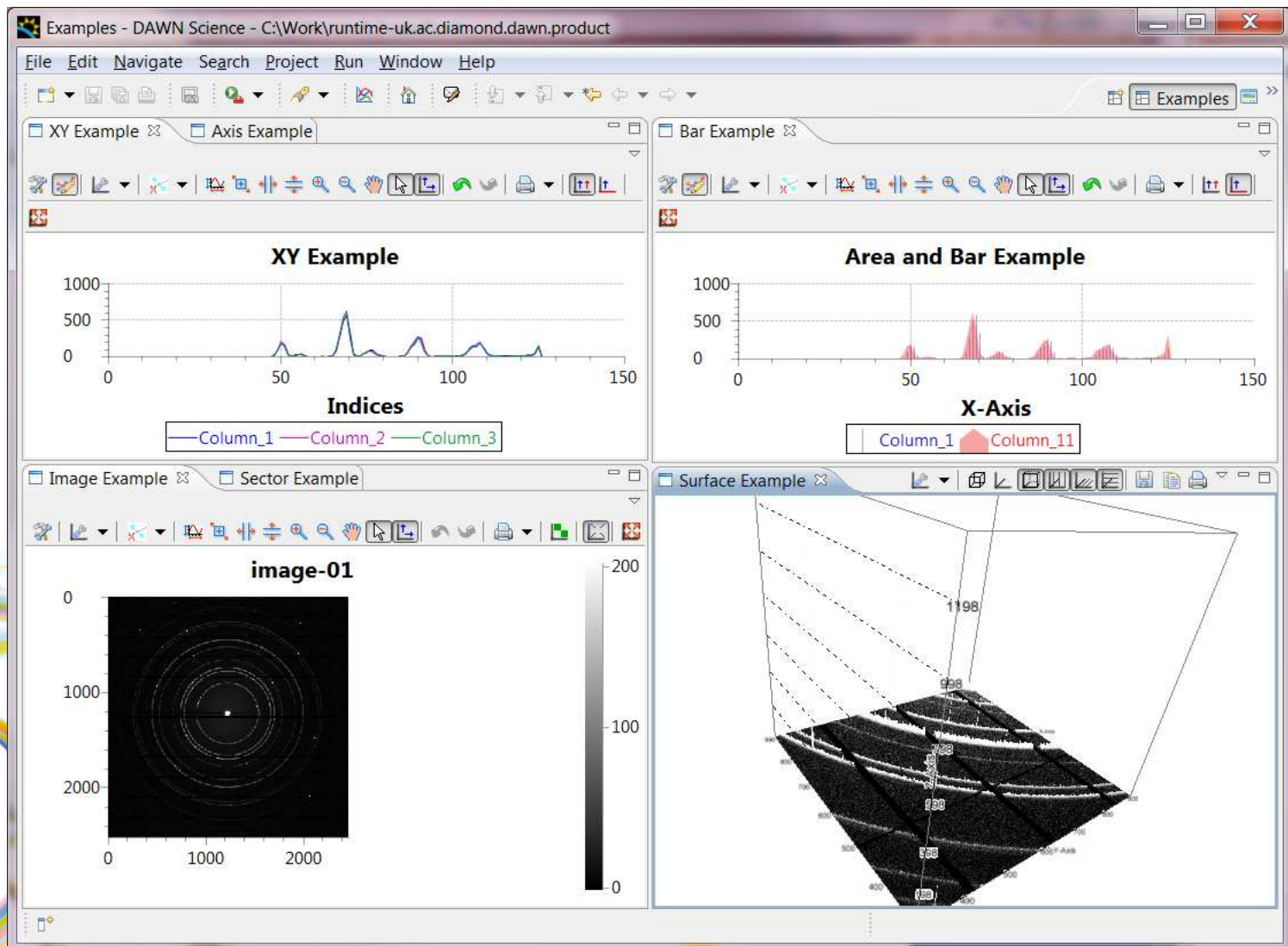
The screenshot displays the DAWNSci GUI interface. On the left is a 'Project Expl' pane showing a file tree with folders like 'workflows', 'data', and 'examples', and various data files such as '315029.dat', '96356.dat', 'i22-4996.nxs', 'image0001.pgm', 'metalmix.mca', 'MoKedge_1_15.dat', 'MoKedge_1_15.nxs', 'pilatus300k.edf', 'pow_M9955_1_0001.cbf', 'ref-screentest-crystal1_1_001.mccc', 'ref-testscale_1_001.img', 'ref-testscale_1_002.img', 'sample1_0050.jpg', and 'spec_scans.spec'. The central area contains a 'Plot' window showing a 2D scatter plot of data from 'metalmix.mca' with axes ranging from 0 to 1000. On the right is a 'Data' pane with a table listing columns and their shapes.

Name	Shape
<input checked="" type="checkbox"/> Column_1	- [128]
<input checked="" type="checkbox"/> Column_2	- [128]
<input type="checkbox"/> Column_3	[128]
<input type="checkbox"/> Column_4	[128]
<input type="checkbox"/> Column_5	[128]
<input type="checkbox"/> Column_6	[128]
<input type="checkbox"/> Column_7	[128]
<input type="checkbox"/> Column_8	[128]
<input type="checkbox"/> Column_9	[128]
<input type="checkbox"/> Column_10	[128]
<input type="checkbox"/> Column_11	[128]
<input type="checkbox"/> Column_12	[128]
<input type="checkbox"/> Column_13	[128]
<input type="checkbox"/> Column_14	[128]
<input type="checkbox"/> Column_15	[128]
<input type="checkbox"/> Column_16	[128]

At the bottom, the 'PyDev Console [1]' shows the following Python code:

```
>>> # Importing scisoftpy.
>>> import scisoftpy as dnp
>>> # Connecting to plot 'metalmix.mca'.
>>> dnp.plot.setdefname('metalmix.mca')
>>> import numpy
>>> Column_1 = numpy.load(r'C:\Users\fc94556\AppData
>>>
```

Demonstration — Examples Plotting and Regions



DAWNSCI - IPlottingService

Purpose – to provide access to plotting without making references to the individual plotting technology in your code.

How – Different plotting systems can be retrieved from the plotting service and are contributed by extension point. DAWNSCI phase 2 will contribute a draw2d / JavaFX based plotting system impl.

Example - Getting a plotting system

```
IPlottingService ps = (IPlottingService)Activator.getService(IPlottingService.class);  
IPlottingSystem sys = ps.createPlottingSystem();
```

```
// This example gets the users current plotting system choice.  
// There are preferences set up by DAWNSci based on each plotting  
// system available / contributed. createPlottingSystem() instantiates  
// and returns the current users preference.
```

DAWNSCI - IPlottingSystem

Purpose – Ability to plot data in 1D, 2D and 3D. Ability to manage selection regions, axes, annotations, printing etc

How – The interface IPlottingSystem is implemented similar to a view part in e3 with a createPlotPart(...) method to add plotting to any SWT composite.

Example - Making a plotting system

```
system.createPlotPart(parent, "Image Example", getViewSite().getActionBars(),
PlotType.IMAGE, this);
IImageTrace image = system.createPlot2D(slice, null, new NullProgressMonitor());

// Plotting system here plots the slice which we previously retrieved.
```

DAWNSCI - IRegion and IRegionSystem

Purpose – Allow selections on the data for algorithms and plotting to use, for instance sectors.

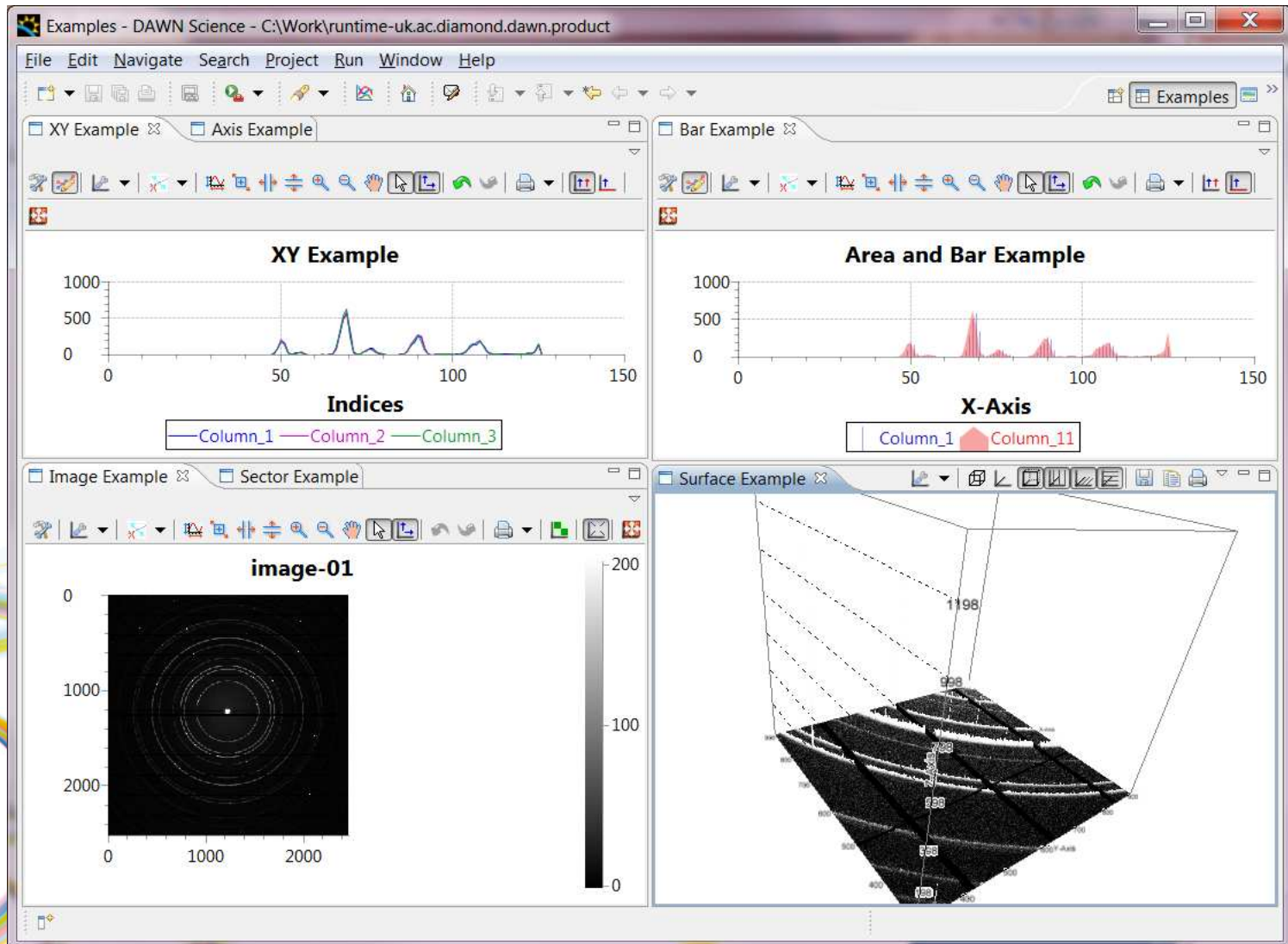
How – The implementation of IPlottingSystem is also IRegionSystem. This allows creation of regions and their configuration/listening. Implemented using draw2d.

```
final IRegion region = sys.createRegion("myRegion", RegionType.BOX);
region.setRegionColor(ColorConstants.orange);
region.setROI(new RectangularROI(0,0,100,100);
region.setMobile(false);
region.setUserObject(FittedFunction.class);
sys.addRegion(region);

region.addROIListener(new IROIListener() {
    // methods for listening to the user moving regions.
})
```


Demonstration — Examples Tools

Look at the plotting examples



DAWNSCI - IToolSystem

Purpose – Encapsulate a set of operations for mathematical interaction with plotting.

How – Generally when using AbstractPlottingSystem a tool system comes for free. Simply return it with the getAdapter(...) method to

Example - Connecting a tool system to the outside world

// In your parts

```
public Object getAdapter(final Class clazz) {  
    if (clazz == IToolPageSystem.class) {  
        return sys.getAdapter(clazz);  
    }  
}
```

```
// Now the external world can see the tool system and tool pages  
// registered by extension point will be available.
```

DAWNSCI - IToolPage

Purpose – Use of IPageView pages to provide helper user interface for a tool linking to a plotting system.

How – Declare with extension point and plot dimensionality and page will appear on plotting systems with registered IToolSystems.



All Extensions

Define extensions for this plug-in in the following section.

- org.eclipse.dawnsci.plotting.api.toolPage
 - Profile (plotting_tool_category)
 - Science (plotting_tool_category)
 - Maths and Fitting (plotting_tool_category)
 - Peak Fitting (plotting_tool_page)
 - Derivative (plotting_tool_page)
 - Derivative View (plotting_tool_page)
 - Measurement (plotting_tool_page)
 - Measurement (plotting_tool_page)
 - Region Editor (plotting_tool_page)
 - XY Information (plotting_tool_page)
 - Pixel Information (plotting_tool_page)
 - Line Profile (plotting_tool_page)
 - Box Profile (plotting_tool_page)
 - Box Line Profile (plotting_tool_page)
 - Perimeter Box Profile (plotting_tool_page)
 - Grid (plotting_tool_page)
 - Radial Profile (plotting_tool_page)
 - Azimuthal Profile (plotting_tool_page)
 - Cross Hair Profile (plotting_tool_page)
 - Zoom Profile (plotting_tool_page)
 - Masking (plotting_tool_page)
 - Circle/Ellipse Fitting (plotting_tool_page)
 - History (plotting_tool_page)
 - Image History (plotting_tool_page)
 - Line Fitting (plotting_tool_page)
 - Diffraction (plotting_tool_page)
 - Function Fitting (plotting_tool_page)
 - Window (plotting_tool_page)
 - Region Sum (plotting_tool_page)
 - Image Normalisation Processing (plotting_tool_page)
 - Image Remapping ARPES Processing (plotting_tool_page)
 - Expression Console (plotting_tool_page)

```

public class ExampleTool extends AbstractToolPage {

    private Composite control;
    public ExampleTool() {
        // Create your listeners to the main plotting
        // Perhaps create a plotting system here from the PlottingFactory which is your side plot.
    }
    @Override
    public ToolPageRole getToolPageRole() {
        return ToolPageRole.ROLE_ID;
    }
    @Override
    public void createControl(Composite parent) {
        this.control = new Composite(parent, SWT.NONE);
        control.setBackground(Display.getDefault().getSystemColor(SWT.COLOR_WHITE));
        control.setLayout(new GridLayout(1, false));
        GridUtils.removeMargins(control);

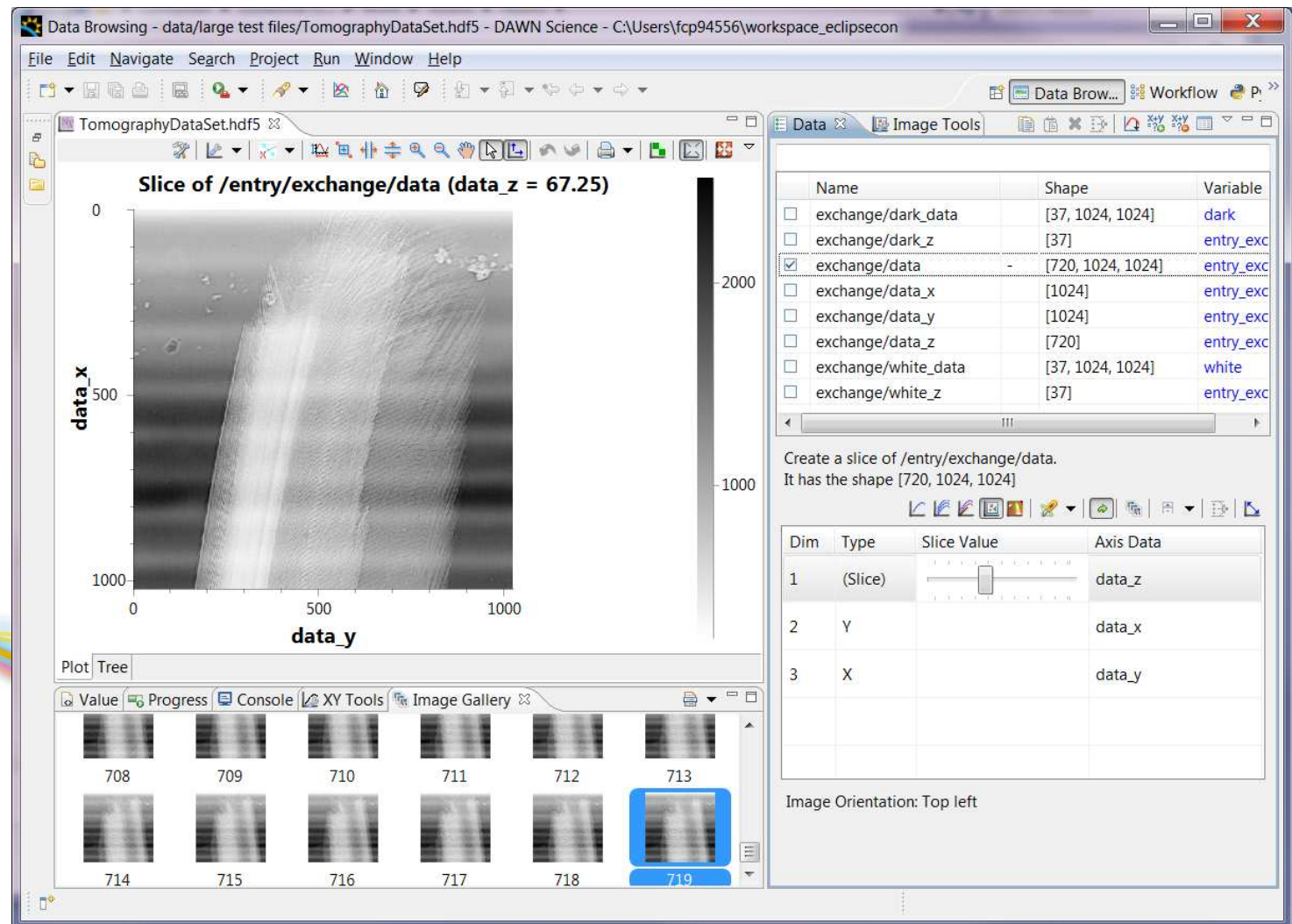
        // User interface shown in a page to the side of the plot.
        // ... For instance: a side plot, a Viewer part
    }
    @Override
    public Control getControl() {
        return control;
    }
    @Override
    public void setFocus() {
        // If you have a table or tree in your tool, set focus here.
    }
    @Override
    public void activate() {
        super.activate();
        // Now add any listeners to the plotting providing getPlottingSystem()!=null
    }

    @Override
    public void deactivate() {
        super.deactivate();
        // Now remove any listeners to the plotting providing getPlottingSystem()!=null
    }
    @Override
    public void dispose() {
        super.dispose();
        // Anything to kill off? This page is part of a view which is now disposed and will not be used again.
    }
}

```

Demonstration — Slicing and dicing

Example opening a tomography file and slicing it



DAWNSCI - ISliceSystem

Purpose – Provide a standardized user interface design to slicing multi-dimensional data.

How – Contributed by extension point, when multi-dimensional data is opened the current active slice system is used to slice the data.

Example - Making a slicesystem

```
public class SliceSystemImpl extends AbstractSliceSystem {  
  
    public Control createPartControl(Composite parent){  
        // ...  
    }  
  
    // Rather a lot of other methods to go into here currently.
```


DAWNSCI - ISlicingTool

Purpose – An extension point registered tool for slicing multi-dimensional data

How – Generally extend AbstractSliceTool and then implement code for doing a slice you need.

Example

```
public class ExampleSlicingTool extends AbstractSlicingTool {  
  
    public void militarize() {  
        //... User chosen the tool  
    }  
  
    public void demilitarize() {  
        //...  
    }  
  
}  
  
// Several examples in the code base
```

DAWNSCI - The Future

- Github move or mirror **dawn-eclipse** to **eclipse/dawnsci**
- Plan for phase 2 – the reference implementation release
 - Cannot do all at once as ~1 million lines
- Release *Isosurface* implementation in 2015 (backed by JavaFX in Java8)
 - Prototype already created summer 2014 and is functional - if you want to try it, get in touch☺
- A **processing pipeline** implementation in 2015.
 - To be used for data reduction at Diamond but in principle any mathematical pipeline can be created and run over stacks

DAWNSCI - Conclusion

- Thanks for listening
- Email us on science-iwg@eclipse.org
- science.eclipse.com is available for new members join now, see Andrew Ross...
- Beverage later?

